523

NPSCS-91-009

NAVAL POSTGRADUATE SCHOOL

Monterey, California

Towards Intelligent Data Retrieval in Mulitmedia Databases

Kyung-Chang Kim

Vincent Y. Lum

February 1991

Approved for public release; distribution is unlimited.

Prepared for:

Naval Ocean Systems Center
San Diego, CA 92152

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. W. West, Jr.
Superintendent

Harrison Shull
Provost

This report was prepared for the Naval Ocean Systems Center and funded by the Naval Postgraduate School.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPSCS - 91 - 009	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Deptartment Naval Postgraduate School 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Ocean Systems Center
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) San Diego, CA 92152
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER OM&N Direct Funding
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Towards Intelligent Data Retrieval in Multimedia Databases

12. PERSONAL AUTHOR(S) Daniel A. Keim, Vincent Y. Lum

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Progress	FROM Oct. 91 TO Jan. 91	February 1991	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Multimedia Database Systems, Information Retrieval, Natural Language Interfaces

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Manipulation of multimedia data in multimedia databases is not straightforward as in conventional databases because of the complex structure of the multimedia data such as image, or sound. The issue in the retrieval of multimedia data from the database is the matching of the contents of multimedia data to a user query. The common solution is to use either keywords or natural language descriptions to describe both the contents of multimedia data and user queries. The major problem is that different users, or the same user at different times, describe the same thing differently which results in the descriptions of the contents of multimedia data to rarely exactly match the descriptions of the user queries. Hence, partial or approximate match between descriptions of multimedia data and user queries is often required during multimedia data retrieval. In this paper, we propose an intelligent approach to approximate match by integrating both object-oriented and natural language understanding techniques.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Vincent Y. Lum	22b. TELEPHONE (Include Area Code) (408) 646-2693 22c. OFFICE SYMBOL CS/Lum

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Towards Intelligent Data Retrieval in Multimedia Databases

Kyung-Chang Kim, Vincent Lum

Computer Science Department[1]
Naval Postgraduate School
Monterey, CA 93943

Abstract

Manipulation of multimedia data in multimedia databases is not straightforward as in conventional databases because of the complex structure of the multimedia data such as image, or sound. The issue in the retrieval of multimedia data from the database is the matching of the contents of multimedia data to a user query. The common solution is to use either keywords or natural language descriptions to describe both the contents of multimedia data and user queries. The major problem is that different users, or the same user at different times, describe the same thing differently which results in the descriptions of the contents of multimedia data to rarely exactly match the descriptions of the user queries. Hence, partial or approximate match between descriptions of multimedia data and user queries is often required during multimedia data retrieval. In this paper, we propose an intelligent approach to approximate match by integrating both object-oriented and natural language understanding techniques.

1. Introduction

A multimedia database management system supports the management of *multimedia data*, which includes image and sound among others, in addition to supporting the conventional databases. Multimedia systems are currently gaining a lot of attention because technology today has made it possible to capture and store in computers multimedia data. Multimedia data broadens the communication between the computer system and the user. Many applications like military, publishing, or instructional routinely need multimedia data. Although the cost of the hardware required to handle multimedia data is decreasing rapidly, the software needed to manage such multimedia data is lacking or inefficient.

[WOELK86] identified two types of requirements which multimedia applications impose on a database system. One requirement is for a data model that allows a natural and flexible definition as well as the evolution of the schema that can represent the composi-

1. This research was supported in part by NOSC and Direct Funding with external sponsor NOSC.

tion of and the complex relationships among parts of a multimedia data. Our earlier work [MEYER88, LUM89] adopted the extended relational model as the basis for addressing all data modelling requirements. The other requirement is for the sharing and manipulation of multimedia data. In this paper, we will emphasize on the sharing and manipulation of multimedia data for multimedia applications.

The focus of this paper is the manipulation of multimedia data in a multimedia database system. In particular, we will describe in detail an efficient method for the retrieval of multimedia data by way of inexact matching. In conventional databases, retrieval of standard numerical and alphanumeric data is handled by utilizing the *content* of the data. The fundamental problem that one must face in the context of a multimedia database is the question of how to handle content search. There is no easy solution because it is difficult to find the appropriate data conveniently and efficiently based on the contents of the multimedia data because they are intrinsically rich in semantics.

In developing an efficient retrieval method for multimedia data, we concluded that it is not possible to utilize the content directly. This is a fair conclusion since the content of a multimedia data is mostly unstructured complex data like an image or a sound. As in most other systems, we follow the approach of content based search by means of verbal descriptions on the contents of multimedia data. A well known keyword approach to content description is not suitable because it has been known to be imprecise and the users often have difficulty in focusing the search to data of interest. Hence, we adopt the natural language approach to content description as a more viable option.

The methodology we adopt consists of associating a natural language description to each multimedia data and using the description to retrieve the relevant data. More precisely, the description of a multimedia data is matched against the description of a user query which is also expressed using a natural language. The major problem with this approach is that it is often the case that the description of a multimedia data does not exactly match the description of a user query. The reason is that it is difficult for different users, or the same user at different times, to describe the same thing identically because they can use synonyms or generalize/specialize categories belonging to the domain of interest and so on. Hence, the key to efficient retrieval process is to automatically perform partial or approximate match of the description of a multimedia data to the description of a user query whenever exact match is not possible. In this paper, we propose an intelligent approach to approximate matching by integrating object-oriented and natural language understanding techniques.

This paper makes two contributions. The major contribution is the formulation of a general scheme to retrieve data that comprises a variety of multimedia data stored in a database with special emphasis on approximate match. As far as we know, very little research on partial or approximate matching has been conducted in natural language processing. The retrieval method may also be adopted easily into the field of intelligent information retrieval. The second related but less significant contribution is to support the claim that ob-

ject-oriented technology can be adopted and applied easily to multimedia systems application.

The paper is organized as follows. Section 2 discusses related work. Section 3 addresses fundamental problems and outlines various components to multimedia data retrieval in a multimedia database system. Section 4 describes in detail our approximate match algorithm during retrieval of multimedia data. The summary is given in Section 5.

2. Related Work

Several multimedia projects have been undertaken by various researchers in both the academia and industry over the past several years. The MINOS system [CHRIST86] developed by a team at the University of Toronto manages highly structured multimedia objects that consist of attributes as well as the text, image and voice part. Sophisticated browsing and user interface features allow browsing of the schema as well as synchronized updates. The MCC Database program [WOELK86,87] also undertook several multimedia projects by establishing the database requirements of multimedia applications. They identified requirements for a data model and for the sharing and manipulation of multimedia data. Hypertext has also been extended to manage image and sound as well. One notable outcome is the INTERMEDIA system [YANKE88] developed at Brown University. [MASU87] has developed a framework to classify and compare the different projects.

In [LWH87] the argument is made that storing multimedia data is one thing but organizing a large amount of them for efficient search and retrieval is quite another. The fundamental difficulty in the retrieval of multimedia data lies in the problem of handling the rich semantics that is contained in the data. In [LUM89], we introduced the approach of contents based search by means of natural language descriptions that form a part of a multimedia data. This approach is related to the research on intelligent information retrieval (IR).

Intelligent IR deals with the overlap of research in artificial intelligence (AI) and IR. Starting several years ago, researchers in information retrieval (IR) became interested in direct access to tabular and textual data. At the same time, researchers in natural language processing are at a point where it can contribute to intelligent IR [SCHA81]. In particular, research in the organization of conceptual memory [DEJON79,KOLO80] may be appropriate to the design of intelligent fact retrieval systems.

A large part of IR research has dealt with techniques for manual and automatic indexing, which is the process of document and request representation. Much of these research is summarized in [SALT74,MARON79]. Also, a strong theoretical framework has been built up in IR around the probabilistic model of retrieval [SALT83,RIJS86]. To overcome the retrieval problem in IR, this model uses pure statistical measures based on keyword frequencies. Finally, clustering research in IR has focused on generating relationships between documents and retrieving groups of documents and has emphasized on solutions

to the best-match problem [SHAS90], also known as the nearest-neighbor problem [SMEA81], or the closest point problem [SHAM75].

There had been early interest of AI techniques in the domain of IR [SPAR78,SMIT80]. The IRUS system [BATES83] is more representative of modern attempts which is designed for processing heterogeneous data bases through natural language queries. The RUBRIC system [TONG87] is a production rule-based IR system in which the indexing base of the system contains positional information about words in the texts, which allow positional controls on words while processing queries. The I^3R system [CROF87] provides assistance to users at all stages of the retrieval process and consists of a set of expert systems managed by a scheduler. Last but not least, the IOTA system [CHIA87] tries to improve the qualitative performance of IR systems in replacing keywords by noun groups involving extensive semantics.

The approach we propose is somewhat different from the intelligent IR systems mentioned. It is clear that most of the work in these systems is mainly concerned with natural language processing, particularly query processing, and deductive capabilities based on extended semantic model of document content and sometimes from the user. Our approach also shares these characteristics. However, the concept of matching function between system concepts and user concepts is based on exact matching in many systems while our approach is based on approximate matching. Even in systems with approximate matching capabilities, the matching function used are primitive or superficial at best compared to our approach which utilizes object-oriented technology to improve the quality of the matching process.

3. Architecture of Multimedia Data Retrieval System

In this section, we outline the architecture of the intelligent retrieval system for a multimedia database system. The architecture consists of the various components of the multimedia data retrieval system. Before we continue, definitions and various issues associated with intelligent retrieval of multimedia data are addressed.

3.1 Definitions and Background

As mentioned before, multimedia data, in the broadest sense, consists of unformatted data such as text, image, voice, signals, etc. in addition to alphanumeric data. We define a multimedia database management system (MDBMS) as a system that manages all multimedia data and provide mechanisms to handle concurrency, consistency, and recovery in addition to providing a query language and query processing.

Despite differences in data model and implementation aspects, all research projects on MDBMS have decided to organize multimedia data using abstract data type (ADT) concept. This is generally accepted as the adequate approach. However, none of the

projects have addressed the problem of content description and retrieval of multimedia data.

The fundamental difficulty in handling multimedia data is intrinsically tied to a very rich semantics. To illustrate such a difficulty, let us look at an image of ships. Given such a picture, how are we to know what type of ships are in the picture. In other words, are the ships destroyers, cruisers, submarines or passenger ships? As another example, let us suppose that there is a picture of a dog and a cat. How do we know if they are chasing each other or playing?

To answer queries posed on images, for example, a person must draw from a very rich experience encountered in life to derive at a good answer. One must have a sophisticated technique to analyze the contents of the images to get the semantics of different things in the images. Technology today is not advanced enough to expect systems to have this kind of capability to answer multimedia query. However, we can use both AI and IR technology to do the next best thing. We can abstract the contents of multimedia data into words or text and use the text description equivalent of the original multimedia data to match the user request or query. This is the principle we will use in designing a MDBMS to handle multimedia data for different applications. Figure 1 shows the format of a multimedia data which consists of the registration, raw and description data.

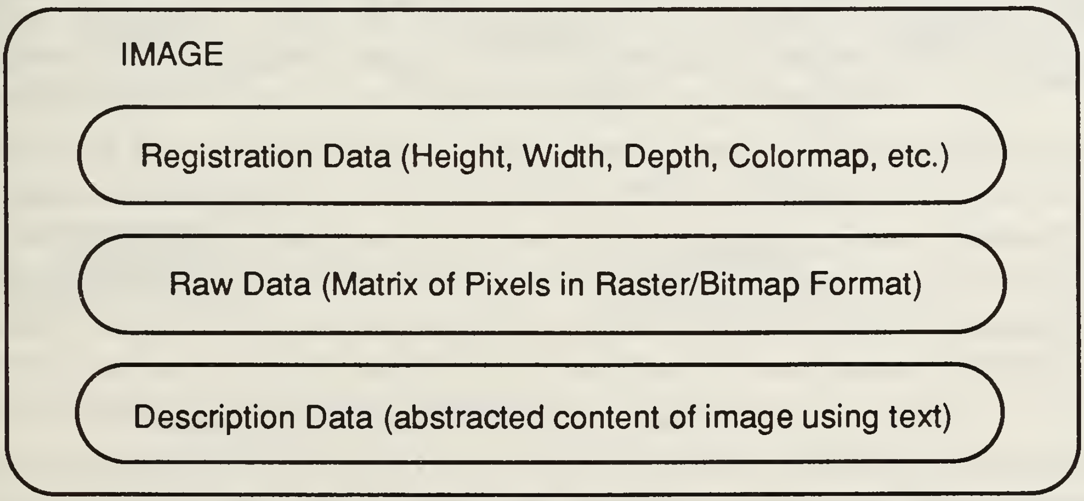

Figure 1: Multimedia Data Format

Raw data is the bit string representation of the image, sound, signal, etc. obtained from scanning or digitizing the original multimedia data. Registration data generally enhances the information about raw data and is not redundant. The contents of a multimedia data is described by description data. Description data cannot be automatically derived by the computer given the technology today. We assume that users will supply the description data for multimedia data in a natural language form.

3.2 Architecture

In this section, we present the various components of a MDBMS that deal with multimedia data retrieval. This is the modified version of the architecture of a MDBMS discussed in [LUM89]. Our proposed architecture enhances the performance of the matcher component and adds the capabilities of the user interface which are lacking in the architecture proposed in [LUM89]. The components of the architecture are shown in Figure 2.

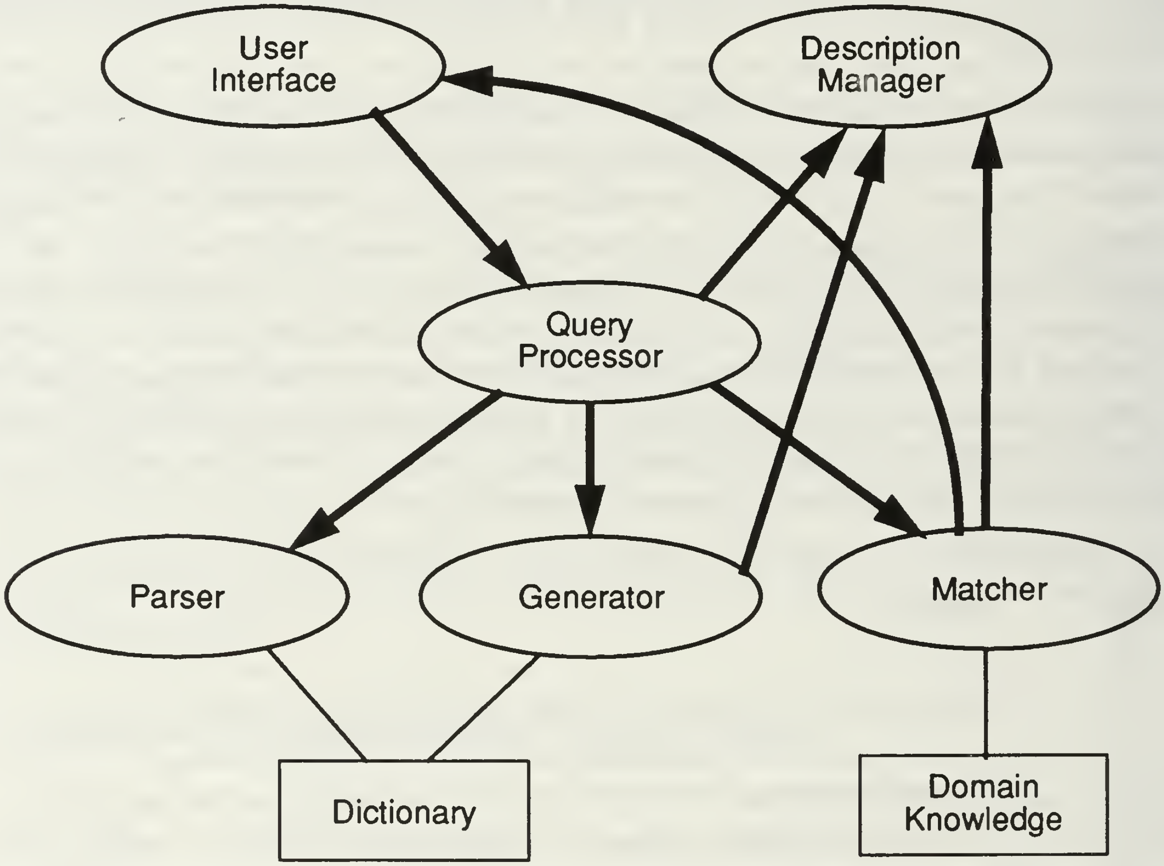

Figure 2: Architecture of MDBMS Data Retrieval System

As shown in Figure 2, the components break down into query processor, description manager, user interface, parser, generator and matcher. The query processor accepts queries from users and executes them by calling the other components. When a new description for a multimedia data is entered, for example, the query processor calls the parser. The parser uses the dictionary to produce first-order predicates and return them to the query processor. The query processor then hands the predicates over to the description manager which then links the description to its multimedia data.

When the query processor receives a query with text description, it calls the parser to obtain the equivalent query predicates. The predicates are then handed to the matcher.

The matcher tries to match the query with the qualified multimedia data by comparing the predicates of the query with that of the stored multimedia data. The matcher does this by calling the description manager and using domain knowledge. In addition, if an exact match is not possible, the matcher automatically switches to approximate match. To guide the matching process, the matcher also gets input from the user by calling the user interface.

As the solution to a query, the query processor returns links to the qualified multimedia data. These links are handed to the generator which calls the description manager to get the predicates. It uses the dictionary to generate a sequence of natural language phrases, which it returns to query processor for output.

The details of each component are discussed in [LUM89, HOLT90]. In addition, the query processor, description manager, parser, matcher and generator have already been implemented as part of the prototype MDBMS developed at the Naval Postgraduate School [MEYER88,LUM89,PEI90]. In this paper, we propose approximate matching process in the matcher as well as the user interaction technique in the user interface.

3.3 Natural Language Description for Multimedia Data

As mentioned, we propose to perform retrieval of multimedia data by matching the natural language descriptions with the query specifications. We discarded the keyword search technique as a viable option because keywords are discrete and lack complex linking mechanisms to adequately capture the contents of multimedia data. In addition, it is not always possible to convey exact meanings using only keywords.

We believe that unrestricted natural language processing is very difficult to achieve given the AI technology today. However, each multimedia application restricts the scope of the description of multimedia data in the particular application. Hence, instead of natural language description, we use captions to describe multimedia data. Captions are a natural but special, stylized way of writing descriptions with a subset of natural language. The details of the captions and their restrictions for our objectives are beyond the scope of this paper and are given in [HOLT90, ROWE91].

Natural language descriptions have the advantage that everyone is familiar with it resulting in high acceptance rate. However this does not solve the problem of description understanding and the matching process as focusing on the specific application domain is also necessary. To handle this problem, a dictionary in which the users define the domain of each application is provided thus restricting their vocabulary, the semantics and the knowledge of the system to apply.

The parser translates the text description into a set of predicates. The imprecision and ambiguity of the natural language descriptions is reduced considerably by transforming them into a set of predicates. These predicates state facts about the real world entities involved with multimedia data like their properties and relationships. As in most parsing

methods, we chose the use of first-order predicate calculus as a formal representation of the description data. The parser depends on the dictionary to turn the descriptions into predicates. It is the parser's task to use the dictionary to resolve synonyms and to check the syntactic context to resolve lexical ambiguities.

Our parser also provides mechanisms to automatically partition a user query into the subject, verb and object components. This is essential in that, during data retrieval as we will see later, we can use the partitioned components to match against domain-dependent knowledge which also break down into subject, noun and object categories. The details of the parser and the predicates are beyond the scope of this paper and are given in [LUM89, HOLT90, DULLE90].

An example of natural language description and its translation into an equivalent set of predicates using the parser is shown below as follows:

Description: "*A car with red body*"

Predicates: *car(x), component(x,y), body(y), color(y,red)*

Choosing the right set of predicates is a very difficult task which is comparable to knowledge acquisition for expert systems. For the purposes of this paper, it is sufficient to assume that the dictionary lists all the words the parser can recognize, all the parts of speech associated with any word, and the predicates to use when a word appears in the description. Thus, the set of all predicates that can be used in the descriptions must be defined in the dictionary.

4. Matching

In this chapter, we propose new ways of matching natural language descriptions of the multimedia data with the query specifications. The key to our matching process is the use of the domain knowledge represented using the notion of class hierarchy borrowed from the object-oriented field. Before we continue, we first discuss some specific problems found in our current matching capability that we eluded earlier. This will serve as the motivation behind our new intelligent approach to approximate matching.

4.1 Problems in Matching

In our current system [LUM89,HOLT90], the result of parsing is one set of predicates per multimedia data instance. A query description is also entered in natural language and parsed. The arguments of the query predicates can be variables. A multimedia data is selected as the result of the query, if there exists a binding of query predicates to description predicates of multimedia data. The match of user query to multimedia data need not be exact. A set of rules, sometimes domain dependent, specifies situations in which sets of predicates that look different are really the same thing.

The matching catches different natural language phrases with the same meaning, but not the semantic relationships among the predicates. For example, let us reconsider the

description, *"a car with red body"*, of an image multimedia data. The predicates generated are *"car(x), component(x,y), body(y), color(y,red)"*. For the sake of argument, we consider a query with the description, *"a red car"*. The query would be translated into something like *"car(x), color(x,red)"*. There would be no match because the system does not know that the color of a car's body is identical to the color of the car.

To overcome this problem, rules can be introduced to express the semantic relationships among the predicates. In the above case, the rule introduced could be:

if *(car(X), component(X,Y),body(Y),color(Y,Z))* **then** *color(X,Z)*;

Using the above rule, color(x,red) can be deduced in the example above and there would be a match between the query and the description. A key unsolved problem, however, is the question of which literals of the predicates to generalize to get a match, and how far to generalize. This falls into the category of approximate matching to a user query that we mentioned earlier in the paper. We believe that the answer lies in the use of domain-dependent knowledge.

If we are just interested in exact matching of a user query to the description of a multimedia data, our current matching technique [LUM89, HOLT90] would be quite adequate. However, a common problem lies in the fact that the user query is likely to result in an empty answer in which no exact matching to the description of stored multimedia data occurs. In this case, an efficient system will try to perform approximate matching whereby descriptions of multimedia data that satisfy some generalization of the user query are selected. Our objective, then, is to perform approximate matching to a user query efficiently. As mentioned earlier, our proposed approximate matching algorithm makes use of domain-dependent knowledge to meet the objective.

4.2 Domain-Dependent Knowledge

Earlier, we justified the use of captions to describe multimedia data by stating that each multimedia application restricts the scope of the description of multimedia data. This means that the domain of discourse for the captions are limited for each multimedia application. Domain-dependent knowledge are key concepts in the domain of discourse of the captions. For our purposes, we only include concepts of nouns and verbs in the domain-dependent knowledge.

To represent domain-dependent knowledge, we chose the object-oriented data model [BANE87, KIM89, ZDON90]. The object-oriented model supports highly structured, complex objects and can capture naturally any mini-world entity. The data model has been used widely in such areas as CAD/CAM, VLSI, office automation, software engineering and AI. Our justification for using the object-oriented model to represent domain-dependent knowledge is as follows: First, it supports generalization and specialization abstraction which permits conceptual generalization on the contents of the captions. Second,

researchers [WOEL87,HOLT90] have identified the use of object-oriented model in multimedia database applications as an appropriate and viable option.

Without loss of generality, we will restrict our domain to the domain of the military history of US forces in the Pacific during World War 2. The main reason is that we tested our current prototype MDBMS in military application based on the domain of the US military history. For our purposes, we will apply our approximate matching technique to the domain of military history. However, we claim that our approximate matching technique can be applied to other multimedia applications.

Figure 3 shows an example of the generalization hierarchy of a plane, a noun concept in our domain of discourse. It is the domain-dependent knowledge on planes that participated in the Pacific during World War 2. We assume that the reader is familiar with object-oriented concepts such as object, class, inheritance along class hierarchy or lattice and methods. We also assume that the direction of the arrow in Figure 3 is from a class to its subclass. In Figure 3, the Plane class is specialized into classes Transport, Fighter, Bomber and Seaplane. Class Transport is specialized into class C-47 and class Fighter is specialized into classes F6F-Hellcat, Corsair and Zero. In addition, class Bomber is specialized into class B-25 and class Divebomber which is further specialized into classes Zero, Dauntless and Stuka.

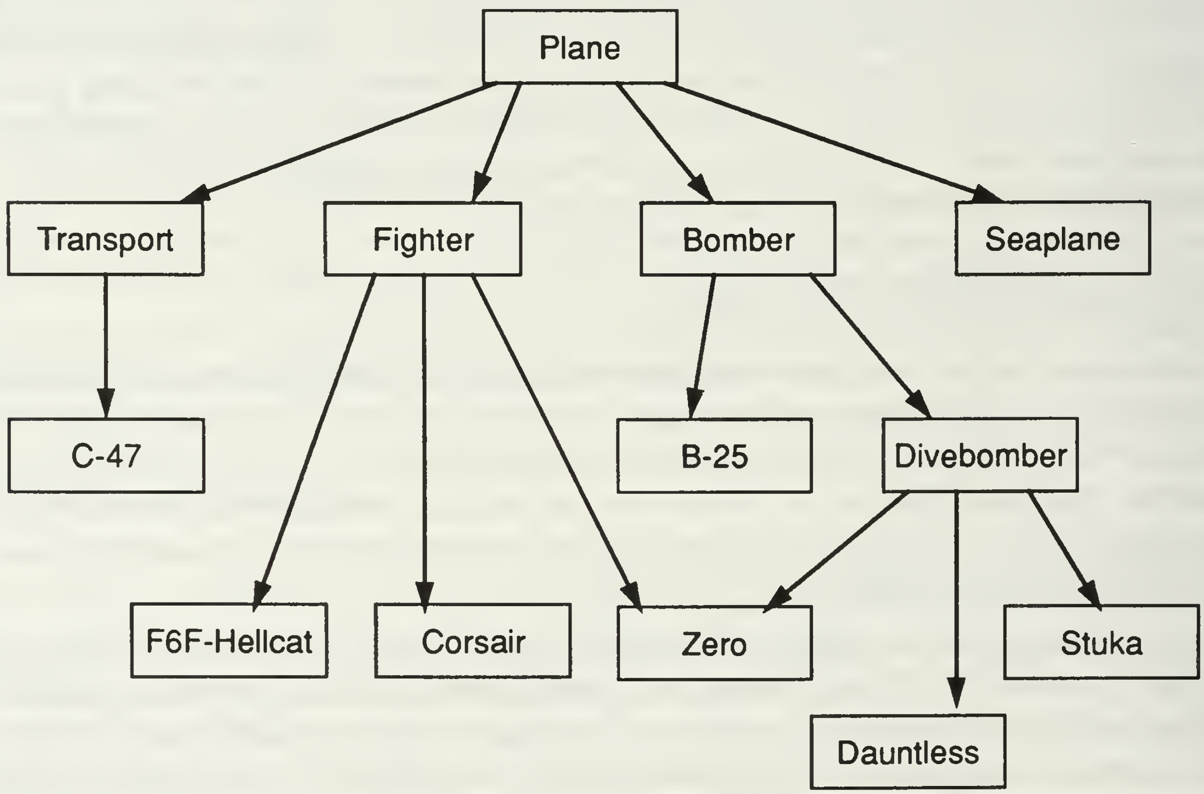

Figure 3: Generalization Hierarchy of a Plane

The generalization hierarchy of a plane is a class lattice since class Zero has two superclasses, namely class Fighter and class Divebomber. In addition, properties of superclasses are inherited by all their subclasses along the superclass/subclass hierarchy but not vice versa.

Figure 3 is one example of a domain-dependent knowledge corresponding to a noun (i.e. plane) concept in the domain of discourse. For our purposes, we can have domain-dependent knowledge for all noun and verb concepts in our domain of discourse. It is obvious that some of the noun and verb concepts may belong to the same class or generalization hierarchy. Hence, generalization hierarchy need not be created for each and every noun or verb concept.

4.3 Partial Matching Algorithm

In this section, we will discuss our partial matching algorithm. For clarity, we will devise our partial matching algorithm by following through an example. Unless explicitly stated, we will refer to the example generalization hierarchy given in Figure 3. Before we go on, we next discuss what it is that we are interested in doing.

Suppose that we have images of planes stored in the multimedia database and the images are described as transport planes. Let us now assume that a user gives a query asking for all planes which are C-47s. Even though there are no exact matching, we should retrieve all transport planes stored because any C-47 is a transport plane according to the domain-dependent knowledge. Now, if the user asks for all fighter planes, we cannot simply retrieve all transport planes because they may not be what the user wants. However, a user asking for planes would more likely retrieve the stored transport planes than if he was to ask for fighter planes because a transport plane is still a plane but is not a fighter plane.

The goal of our algorithm is also to minimize the influence of the definition of the hierarchy which is dependent on the designer. The generalization hierarchy designer might have a view of the domain dependent knowledge which may not be consistent with the view of other people. This phenomenon might bias some specific branch of the generalization hierarchy over other branches during partial matching.

An efficient partial matching algorithm has to deal with all the problems such as the ones addressed above and come up with a general solution. We solve these problems by using heuristics to assign a weight ranking system given a generalization hierarchy(ies). Our major objective is to come up with a weight ranking scheme that is both fair and accurate which can be used to determine whether stored multimedia data should be retrieved given a user description.

4.3.1 Weight Ranking Scheme within a Generalization Hierarchy

In this section, we will discuss the weight ranking strategy used by our partial matching algorithm given a single generalization hierarchy. The weight ranking strategy used for a group of generalization hierarchies will be discussed in the subsequent section. The weight ranking strategy used on a generalization hierarchy is a consequence of the semantics of the class hierarchy (lattice) or the IS-A hierarchy concept supported in an object-oriented data model.

Given a class C in a generalization (class) hierarchy for a noun or a verb concept, and assuming that a class, other than C, with a rank of positive weight is a specialization of class C while one with a rank of negative weight is a generalization of C, we can introduce the following two general heuristics;

Heuristics 1 : *All direct (indirect) subclasses of C have positive weights.*

Heuristics 2 : *All direct (indirect) superclasses of C have negative weights.*

Heuristic 1 says that given a class C specified in a user query, all subclasses of C in the class hierarchy to which C belongs are specializations of the class and more weights (positive) are given. Heuristics 2 says that given a class C specified in a user query, all superclasses of C in the class hierarchy to which C belongs are generalization of the class and less weights (negative) are given. This reasoning follows directly from the definition of a class (IS-A) hierarchy and relationships among classes along the class hierarchy in the context of an object-oriented data model.

The assignment of negative weights to generalization is intuitively clear. The assignment of positive weights to specialization is based on the fact that specialization inherits all properties of the parent nodes in addition to having its own additional information. Hence, we feel that positive or more weights should be assigned to the nodes in the paths towards specialization hierarchy.

Given the heuristics, it is easy to see that all classes in the class hierarchy which have ranks of positive weights relative to the class C, which is specified in the user query as either a noun or a verb concept, are selected during approximate matching. This is because all classes with ranks of positive weights are subclasses (specialized classes) of the class C, specified by the user query. Since each of the classes is a specialized version of class C, it encompasses properties of class C and indeed is class C.

On the other hand, all classes in the class hierarchy which have ranks of negative weights relative to the class C which is specified by the user query should be restrictively selected depending on the weights. This is because all classes with ranks of negative weights are superclasses (generalized classes) of the specified class C along the class hierarchy. Since each of the classes is a generalized version of class C, it does not encompass all properties of class C and is not class C. The question of which classes to select depends on getting information from the user on how far to generalize.

The weight ranking system we introduced so far is vague and is not well defined. What is defined is that given a class C in a class hierarchy of interest, any class belonging to the same class hierarchy which is assigned a positive weight is always selected during approximate matching. On the other hand, a class in the same class hierarchy which is assigned a negative weight is only selected during approximate matching if it exceeds a threshold given by the user. We now discuss the assignment of weights for different classes in the class hierarchy of interest.

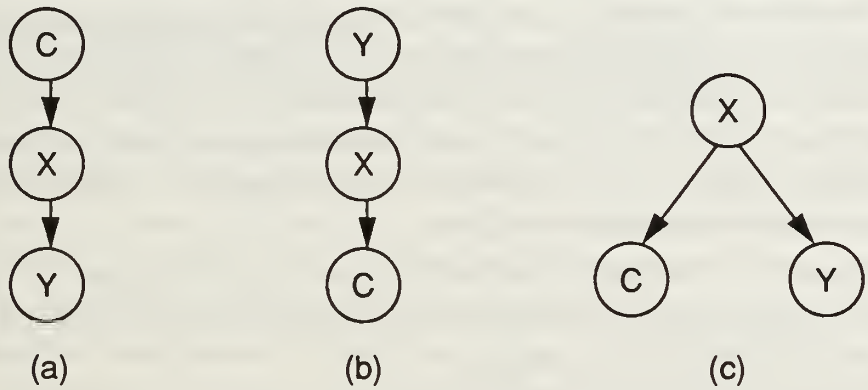

Figure 4: Three Situations in a Class Hierarchy

There are three different situations in which weights can be assigned to classes in a class hierarchy. The different situations are shown in Figure 4. Suppose that the class specified in a user query is class C. As before, we assume that the direction of the arrow is from a class to its subclass. For example, in Figure4 (a), class C is a superclass of class X and class X is a subclass of class C. The first situation, shown in Figure 4(a), is to assign weight to a class (X or Y) which is a subclass of class C. The second situation, shown in Figure 4(b), is to assign weight to a class (X or Y) which is a superclass of class C. The third situation, shown in Figure 4(c), is to assign weight to a class (Y) which is a subclass of a superclass (i.e. X) of class C.

The principles behind our weight ranking system are quite simple. We assume that all classes with positive weights and some classes with negative weights that exceed a *threshold value* are selected during approximate matching. First, we assign a weight of *0* to the class C specified in the user query. Class C is the reference point to all other classes in the class hierarchy during approximate matching. For classes which are subclasses of class C, we assign *positive weights* because they are specialized version of class C. Specialized versions of class C have more specific and definite information than C itself and hence are assigned positive weights instead of 0. For our purposes, all subclasses of C are assigned the same positive weight.

For classes which are superclasses or subclasses of superclasses of class C, we assign *negative weights* because they are generalized version of class C. Generalized ver-

sions of class C have less and more general information than C itself and hence are assigned negative weights. Different generalization versions have different negative weights. However, in assigning negative weights, we have to minimize the influence of the definition of the model. It is true that the further away a class is from class C in the class hierarchy, the more negative weight is assigned to the class.

In most systems, the assignment of weight of a class is linearly inverse proportional to the depth level of the class relative to the level of the class C specified in the user query. We believe that this is not the correct approach because the relative distance of a particular class to the class of interest, in this case class C, with respect to other classes is not the absolute but some artificial distance caused by a particular designer's view of the domain knowledge. The main problem with this approach is that some classes belonging to some lengthy branch could be unfairly disqualified because of higher negative weights. Our weight ranking system tries to minimize the bias against some lengthy branch of a class hierarchy over other shorter branches.

Given that class C is the class specified by user query as shown in Figure 4, the assignment formulas of weights for classes in a class hierarchy according to the three different situations mentioned are as follows.

(1) Class specified by user query (i.e. class C in Figure 4)

$$\mathbf{weight} = 0$$

(2) Subclass of class C (i.e. class X or Y in Figure 4(a))

$\mathbf{weight} = \alpha$, where α is a integer constant

(3) Superclass of class C (i.e. class X or Y in Figure 4(b))

$$\mathbf{weight} = -\left(\alpha \times \sum_{i=1}^{n} \left(\frac{1}{\beta}\right)^{i}\right),$$

where α, β are integer constants and n is level # of superclass relative to class C

(4) Subclass of a superclass of class C (i.e. class Y in Figure 4(c))

$$\mathbf{weight} = -\left(\alpha \times \sum_{i=1}^{h} \left(\frac{1}{\beta}\right)^{i}\right) - \left(\gamma \times \sum_{j=1}^{l} \left(\frac{1}{\upsilon}\right)^{(l+1-j)}\right),$$

where α,β,υ,γ are integer constants; h is level # of superclass relative to class C and l is level # of subclass relative to superclass

In our scheme, a class which is assigned a positive weight is always selected during partial matching. A class with a negative weight can be selected provided that it does not exceed a threshold value set by the user. To understand the weight assignments for different classes, we next give some examples using the class hierarchy of Figure 3. Given a user query, if the image corresponding to the user description is not found in the database, the system then automatically proceeds with approximate matching. Using the

weight assignment formulas and given some user query descriptions, the weights for some of the classes in the class hierarchy are as follows. For the sake of argument, we assume that the values of α, β, γ and υ are 40, 2, 48 and 2 respectively.

(1) **"A transport plane sank in the Pacific"**
Transport = 0, C-47 = 40, Plane = -20, Fighter = -44, Corsair = -56

(2) **"A F6F-Hellcat sank in the Pacific"**
F6F-Hellcat = 0, Plane = -30, Seaplane = -54, Stuka = -72, C-47 = -66

(3) **"A bomber sank in the Pacific"**
Bomber = 0, Stuka = 40, B-52 = 40, Plane = -20, Seaplane = -44, C-47 = -56

In the examples shown, all classes which are assigned positive weights are selected during partial matching. In example (1), the class C-47 has a positive weight of 40. This means that the image whose description is "**A C-47 sank in the Pacific**" is selected during partial matching. As shown in the examples, all classes which are subclasses of the class which is specified in a user query are assigned positive weights. All classes which are superclasses or subclasses of the superclasses of the class which is specified in a user query are assigned negative weights. For these classes, the weight of a class is inversely proportional to the depth level of the class relative to the level of the class specified in the user query along the class hierarchy although they are not strictly linear. In example (2), the class Seaplane has a negative weight of -54. This means that the image whose description is "**A Seaplane sank in the Pacific**" has a weight of - 54. Class Stuka has a negative weight of -72 and class Stuka is further away from F6F-Hellcat than class Seaplane is from F6F-Hellcat.

Suppose the weight of a class is linearly but inversely proportional to the depth level of the class relative to the level of the class C specified in the user query. If we assign a negative constant weight, say -10, for each level away from class C, the class which is 5 levels away from class C will have a negative weight of -50 compared to a negative value of -20 for a class which is 2 levels away from class C. For example, if the user query is "**A transport sank in the Pacific**", the weight of class Transport is 0, class Seaplane is -20 and class Stuka is -40. Using our formulas, the same user query will assign weights of classes Transport, Seaplane and Stuka to be 0, -44 and -62 respectively. The weight of class Stuka is more biased against relative to the weight of class Seaplane using the linear method over our method.

It is very difficult to quantify how much closer class Seaplane is to class Transport over class Stuka to class Transport as both Seaplane and Stuka are types of planes. Our formulas are designed to minimize bias as best as possible. A user is more likely select a threshold value such that class Stuka is less likely selected during approximate matching over class Seaplane using the linear method compared to using our dynamic method. Another difficult task is to set the value of the constants to be applied in our assignment for-

mulas as well as the threshold value. The user must choose the correct values for the constants and the threshold value depending on the number of objects that qualify during approximate matching. Hence, it is necessary for the system to interact with the user through the user interface throughout the matching process.

4.3.2 Weight Ranking Scheme for a Group of Generalization Hierarchies

In the previous section, we discussed the ranking of weights for classes belonging to the same generalization hierarchy. In this section, we extend the ranking of weights for classes belonging to different generalization hierarchies. In our scheme, the ranking of weights for classes in different class hierarchies is influenced by the following rules.

Rule 1: *For each local class hierarchy, the weight ranking system discussed in Section 4.3.1 is applied.*

Rule 2: *For different class hierarchies, the user determines the priority order of class hierarchies.*

Using rule 1, for each class hierarchy selected by the user query, the classes within the class hierarchy are assigned weights using the weight ranking system discussed in Section 4.3.1 and is a straightforward process. Hence, regardless of the number of class hierarchies involved, all classes belonging to these hierarchies can be assigned weights for partial matching. It is easy to see that rule 1 does not cause any problems because there is no interrelationship between classes of different class hierarchies during weight assignments.

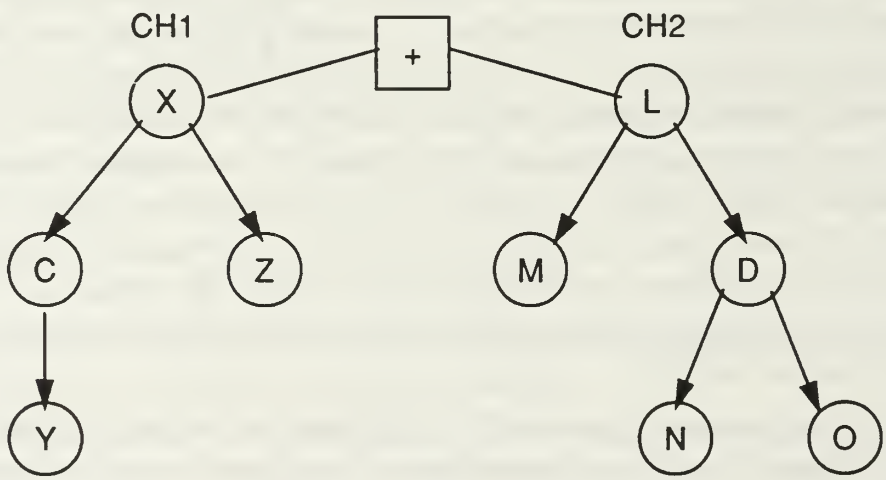

Figure 5: Combination of two Class Hierarchies

The global ranking of weights for different class hierarchies is a problem because the weights assigned within each class hierarchy now has to be considered with respect to weights assigned for other class hierarchies and they have to be meaningful globally. Rule 2 is to determine the priority order of importance of the class hierarchies selected from a user through the user interface. Different class hierarchies can be assigned different weights according to the priority order of importance.

Figure 5 shows a combinations of classes belonging to two different class hierarchies. For our purposes, we now consider a user query description **"C and D"** involving classes C and D belonging to different class hierarchies in Figure 5. There are three generic partial matching combination types and they are given as follows. The example classes in the combination types are taken from Figure 5. CH1 is the name of the class hierarchy on the left and CH2 is the name of the class hierarchy on the right in Figure 5.

Type 1 : C of CH1, and any class in CH2 except D.

Type 2 : Any class in CH1 except C, and D of CH2.

Type 3 : Any class in CH1 except C, and any class in CH2 except D

The ranking of weights for type 1 and type 2 combinations are easy to handle by using the previously discussed weight ranking system. This is because we only need to assign weights to classes in one of the class hierarchies but not both. However, handling type 3 combination requires a closer attention because it requires assigning weights to classes belonging to different class hierarchies. To assign weights in this case, we determine the priority order of CH1 and CH2 through feedback from the user. Through the user interface, we get information on which class hierarchy has a higher priority. We then assign different weights for CH1 and CH2 depending on the priority order.

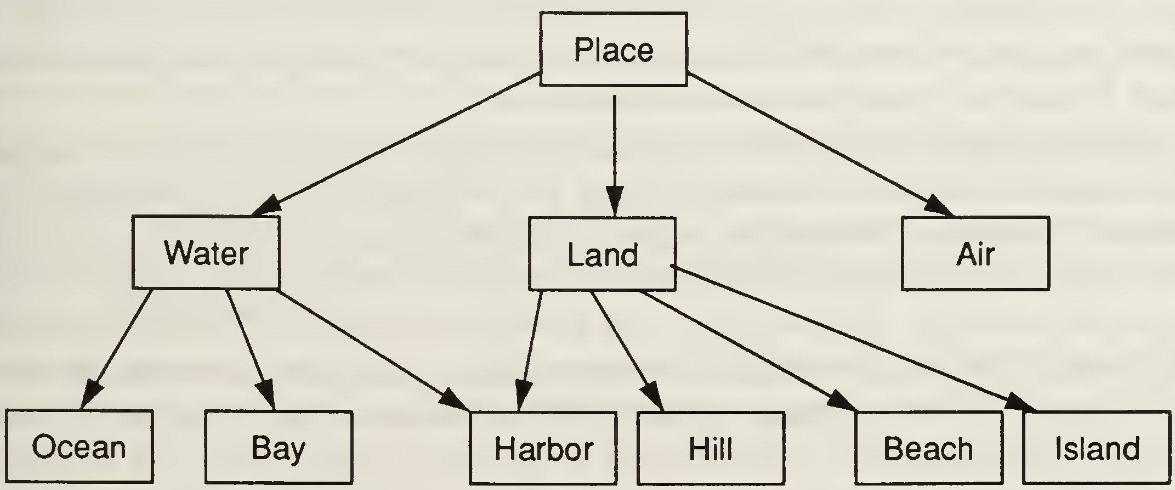

Figure 6: Generalization Hierarchy of a Place

This can be expressed using the following weight formula. The constant values of ω and δ has to be determined by the user through the user interface.

(1) **Weight (Type 1) = Weight (CH2)**

(2) **Weight (Type 2) = Weight (CH1)**

(3) **Weight (Type 3) = ω (Weight (CH1)) + δ (Weight (CH2))**

Figure 6 is a generalization (class) hierarchy of the noun Place concept. Using Figure 3 and Figure 6, given a user query description **"A C-47 sank in the Ocean"**, an example of a type 1 combination is a multimedia data with a description **"A C-47 landed in an Island"**. A type 2 combination is a multimedia data with a description of **"A Stuka sank in the Ocean"**. Finally, a type 3 combination is a multimedia data with a description **"A Transport landed in Sea"**.

In this section, we discussed the assignment of weights for classes involving two different class hierarchies. For a practical system, the number of class hierarchies involved for weight assignment is obviously large since many noun and verb concepts are involved. It is not difficult to see that our weight ranking scheme discussed in this section can be easily extended to assign weights for classes involving many class hierarchies. The main problem lies in how good the user interface is in getting the information from the user. Obviously, the weight ranking system has to be dynamic, since all constant values assigned by the user can change depending on the number of qualified multimedia data selected during partial matching. The user also has to determine the threshold value such that not too many multimedia data are selected from the database.

4.3.3 Application of Weighting Algorithm

The application of the weighting algorithm just presented requires a parser to understand the natural language specifications in the multimedia data descriptions and the user queries. As stated earlier, the descriptions are parsed and stored in the system as predicates. The queries are processed as follows.

When a query is received from the user, the parser separates the natural language specification into smaller component groups, namely subject noun, verb and object noun phrases. Each of these will actually become predicates. When these predicates match exactly with the predicates in the descriptions of certain multimedia data, those multimedia data will be retrieved. However, there may be other descriptions of multimedia data that are actually of interest to users but those descriptions are not stated as logically implied by the query. This latter category is expected to be the usual case rather than the former for reasons stated earlier.

To find the latter, we suggest that system search in the noun and verb generalization hierarchies of the object classes and assign weights to the descriptions as given in the weight assignment algorithm, assigning the appropriate weighting factors (ω and δ in the

previous section) as received from the user. These multimedia data with combined weight exceeding the threshold value set by the user will then be retrieved.

The separation of the natural language query can be in smaller components than the three groups just stated. For example, a complex noun phrase may be separated into a number of small noun groups and the weighting algorithm applied to these groups to obtain a combined weight. For example, "*the man with a mustache*" can become two classes, namely man and mustache. Naturally, the finer the granularity of the separation, the larger and the more complex the processing is needed.

5. Summary

A major problem in retrieving multimedia data such as a sound, or an image stored in a database is that they are intrinsically rich in semantics and conventional search methods used in databases and information retrieval systems may not work or are of little use. Most research on intelligent IR systems are concerned with natural language processing and deductive capabilities based on extended semantic model of document content and also from the user. However, most of them deal with exact matching or primitive partial matching using simple linear methods.

In this paper, we discussed some of the fundamental problems faced by a MDBMS during data retrieval and outlined an architecture of a IR system for multimedia databases. The main contribution of our paper is the formulation of a partial matching algorithm that uses domain knowledge, represented using an object-oriented data model, and weight ranking system to assign weights to different multimedia data stored in a database and selects those multimedia data that partially matches a given user query description. Our dynamic matching scheme also interacts with the user to get further information regarding the soundness and correctness of the partial match.

Our parser provides mechanisms to automatically partition a user query into the subject noun, verb and object noun components. This is essential in that, during data retrieval, we used the partitioned components to match against generalization hierarchies of domain-dependent knowledge which also deals with noun and verb categories. Further research is necessary to improve the parser to also automatically derive adjectives and other caption components for complete understanding and processing of captions in the context of partial matching.

We believe that our approach is both simple and elegant. The simplicity lies in exploiting the semantics of generalization and specialization abstraction of the object-oriented model. We also believe that our approach is a general one that can be readily applied to applications in IR and AI. However, in this paper we left out details of the user interface component of our retrieval system. We believe that this is an area of research interest and we are currently investigating different methods.

References

[BANE87] Banerjee, J. et. al., "Data Model Issu4es for Object-Oriented Applications," ACM TOOIS, January 1987

[BATES83] Bates, M. and B. Bobrow, "Information retrieval using a transportable natural language interface," Proc. of the 6th. ACM SIGIR conf. on R&D in Information Retrieval, Bethesda, MD, 1983

[CHIA87] Chiaramella, Y. and B. Defude, "A prototype of an intelligent system for information retrieval: IOTA," Information Processing and management, Vol. 23, No. 4, pp. 285-303, 1987

[CHRIS86] Christodoulakis, S. et. al., "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System, " ACM TOOIS, Vol. 4, no. 4, Oct. 1986, pp. 345-383

[CROFT87] Croft, W. and R. Thompson, "I^3R: A New Approach to the Design of Document Retrieval Systems," Journal of the American Society for Information Science, 38, pp. 389-404, 1987

[DEJON79] Dejong, G.F., "Prediction and substantiation: A new approach to natural language processing," Cognitive Science 3, 3 (July 1979), 251-273

[DULL90] Dulle, J. "The Scope of Descriptive Captions for Use in a Multimedia Database System," M.S. Thesis, Computer Science Department, Naval Postgraduate School, Monterey, CA, June 1990

[GOLD85] Goldman, K. J. et al. "ISIS: Interface for a Semantic Information System", Proc. ACM-SIGMOD 1985 Int'l Conf. on Management of Data, Austin, Texas, May 1985, pp. 328-342

[HOLT90] Holtkamp, B. et. al., "Demon- A media object model incorporating natural language descriptions for retrieval support," Tech. Report NPS52-90-019, Computer Science Department, Naval Postgraduate School, Feb. 1990

[KIM89] Kim, Kyung-Chang et. al., "Cyclic Query Processing in Object-Oriented Databases," Proc. 7th. Int'l Conf. on Data Engineering, L.A., CA, Feb. 1989

[KOLO80] Kolodner, J.L., "Organizing memory and keeping it organized," Proc. ist. Annual Nat. Conf. Artificial Intelligence, Aug. 1980, pp. 331-333

[[LUM89] Lum, V. and K. Meyer-Wegener, "A Multimedia Database Management System Supporting Content Search in Media Data," Tech. Report NPS52-89-020, Computer Science Department, Naval Postgraduate School, Monterey, CA, March 1989

[LWH87] Lum, V. et. al., "Integrating Advanced Techniques into Multimedia DBMS," Tech. Report NPS52-87-050, Computer Science Department, Naval Postgraduate School, Monterey, CA, Nov. 1987

[MARON79] Maron, M.E., "Depth of indexing," J. Am. Soc. Inf. Sci., 30 (1979) 224-228

[MASU87] Masunaga, Y. "Multimedia Databases: A Formal Framework," Proc. IEEE CS Office Automation Symp. IEEE CS Press, order no. 770, Washington

1987, pp. 36-45

[MEYE88] Meyer-Wegener, K. et. al., "Managing Multimedia Data," Tech. Report NPS52-88-010, Computer Science Department, Naval Postgraduate School, Monterey, CA, March 1988

[PEI90] Pei, Su-Cheng "Design and Implementation of a Multimedia DBMS : Catalog Management, Table Creation and Data Insertion," M.S. Thesis, Computer Science Department, Naval Postgraduate School, Dec. 1990

[RIJS86] Van Rijsbergen, C.J., "A New Theoretical Framework for Information Retrieval," Proc. ACM conf. on R&D in information retrieval, pp. 194-200, Sept. 1986

[ROWE91] Rowe, N. and E. Guglielmo, "Exploiting Captions for Access to Multimedia Databases," To appear in IEEE Computer , 1991

{SALT74] Salton, G. et. al., "Contributions to the theory of indexing," Information Processing 74, Elsevier-North Holland, New York, 1974 584-590

[SALT83] Salton, G. and M.J. McGill, "Introduction to information retrieval," McGraw-Hill, Inc., New York, NY, 1983

[SCHA81] Schank, R. et. al., "Conceptual information retrieval," Information Retrieval Research, Oddy, Robertson, van Rijsbergen, Williams, Eds., London, 1981

[SHAM75] Shamos, M. I. and D. Hoey, "Closest-point problems," In Proc. of the 16th. IEEE Symposium on FOCS (Oct. 1975), pp. 151-162

[SHAS90] Shasha, D. and Tsong-Li Wang, "New Techniques for Best-Match Retrieval," ACM TOIS, Vol. 8, No. 2, April 1990

[SMEA81] Smeaton, A.F. and C.J. Van Rijsbergen, "The nearest neighbor problem in information retrieval: An algorithm using upperbounds," ACM SIGIR Forum 16 (1981), pp. 83-87

[SMIT80] Smith, L.C., "Artificial intelligence applications in information retrieval," Annual Revue on Information Science Technology, 15:67-115, 1980

[SPAR78] Spark, K. Jones, "Artificial intelligence: what can it offer to information retrieval," Proc. of the Informatics 3, Aslib, ed., London, 1978

[TONG87] Tong, R. et. al., "Conceptual Information Retrieval using RUBRIC," Proc. ACM SIGIR Conference, pp. 247-253, 1987

[WOEL86] Woelk, D. et. al., "An object-oriented Approach to Multimedia Databases," Proc. ACM SIGMOD, May 1986

[WOEL87] Woelk, D. and W. Kim, "Multimedia Information Management in an Object-Oriented Database System, " Proc. of VLDB, Brighton, England, Sept. 1987

[YANK88] Yankelovich, N. et. al., "Intermedia: The Concept and the Construction of a Seamless Information Environment," IEEE Computer, Vol. 21, No. 1, Jan. 1988, pp. 81-96

[ZDON90] Zdonik, S. and D. Maier, "Readings in Object-Oriented Database Systems," Morgan Kaufmann Publishers, Inc. 1990

Distribution List

SPAWAR-3242 Attn: Phil Andrews Washington, DC 20363	1 copies
Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145	2 copies
Library, Code 52 Naval Postgraduate School, Monterey, CA 93943	2 copies
Center for Naval Analyses, 4401 Ford Avenue Alexandria, VA 22302-0268	1 copy
Director of Research Administration, Code 08, Naval Postgraduate School, Monterey, CA 93943	1 copy
John Maynard Code 402 Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1 copy
Dr. Sherman Gee ONT-221 Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000	1 copy
Leah Wong Code 443 Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1 copy
Vincent Y. Lum Naval Postgraduate School, Code CS, Dept. of Computer Science Monterey, California 93943-5100	25 copies

Daniel A. Keim
Comupter Science Department
University of Munich
Theresienstr. 39
D-8000 Muenchen 2
Germany 5 copies

Kyung-Chung Kim
Computer Science Department
Hong-Ik-University
72-1 Sangsu - Dong, Mapo-Ko
Seoul, Korea 1 copy

Neil C. Rowe
Code CSRp
Naval Postgraduate School
Monterey, CA 93943 1 copy

Thomas C. Wu
Code CSWu
Naval Postgraduate School
Monterey, CA 93943 1 copy

Bernhard Holtkamp
University of Dortmund
Software Technology Lab
Postfach 500500
D-4600 Dortmund 50
Germany 1 copy

Klaus Meyer Wegener
University of Erlangen
Computer Science Department
P.O. Box
Erlangen
Germany 1 copy